Paper AD01

Validating a CDISC Dataset at development time using Pinnacle 21 CLI via SAS

David Meek, PHASTAR, Glasgow, United Kingdom

ABSTRACT

Phastar have developed a macro called **%p21_validate** to run Pinnacle 21® Community validator CLI (Command Line Interface) direct from SAS® 9.4 using the X command to validate an individual CDISC dataset at development time and provide real-time validation feedback in the SAS output window to catch validation issues early in the development cycle.

INTRODUCTION

CDISC dataset validation issues are often caught too late in the study development cycle. A minor update at ADaM level prior to a Sponsor delivery of TLFs can require a programming update tracking back to SDTM resulting in updates for minor compliance issues, which then in turn require a time-consuming re-run of TLF code to ensure the correct run order of dependencies for both production and QC.

This paper covers individual dataset validation using Pinnacle 21 Community validator versions 2.2.0 and 3.0.1 in a Microsoft Windows® environment. Using the X command in SAS allows us to submit a Windows command without ending our SAS session which then allows us to run the P21 validator application and return the results to SAS.

Individual dataset validation is intended as an extra check at programming development time, but once available all study datasets should be validated together as a package in P21 during the study to ensure cross-check validation for complete compliance.

KEY ADVANTAGES

By running the validator from within SAS during development time, compliance issues can be flagged immediately for a programmer to fix. It is also more time efficient to identify issues and correct specification files or investigate data issues at dataset development time rather than retrospectively or at the end of the study.

P21 JAR LOCATION

To work with **%p21_validate**, we need to know where the P21 CLI JAR (Java ARchive) file is stored to include in our SAS code. A JAR file is a package file that contains Java programs and related files in a single file.

For 2.2.0, the CLI JAR is stored here within the chosen installation folder:

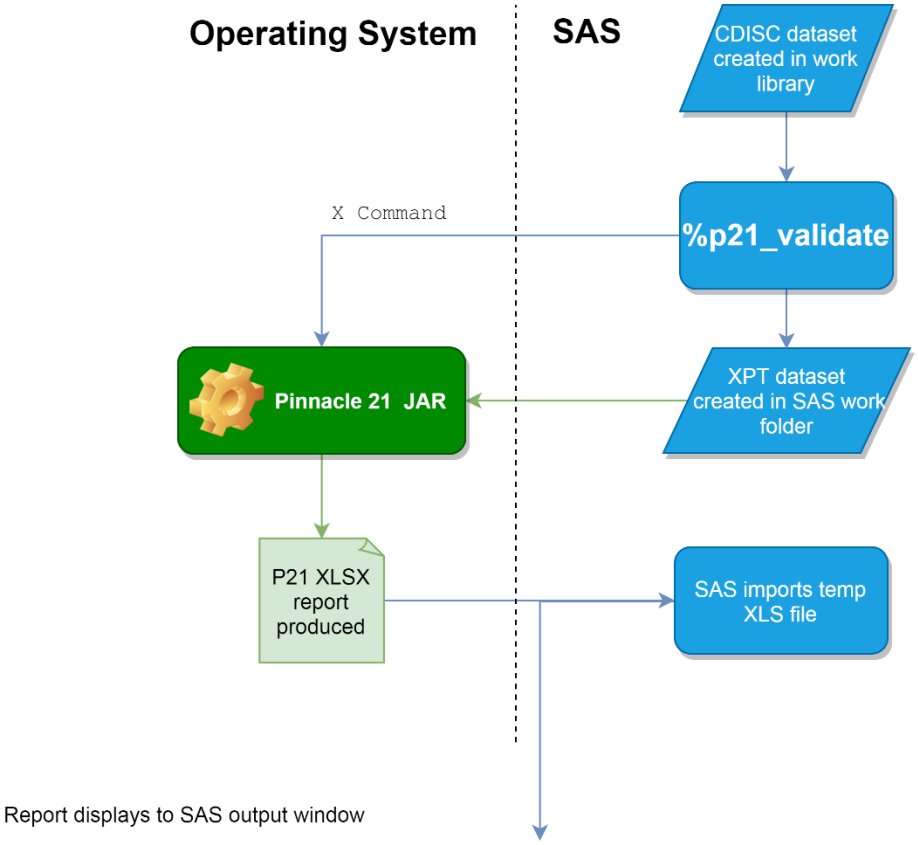
```
\\...\tools\pinnacle21-community-2-2-0\components\lib\validator-cli-2.1.5.jar
```

For 3.0.1, the CLI JAR is stored within the following path after installation:

```
C:\Users\%username%\AppData\Local\Programs\pinnacle21-community\resources\app.asar.unpacked\components\lib\p21-client-1.0.1.jar
```

To run the JAR, we need to be able to reference those JAR files. For 2.2.0, the JAR is a stand-alone validator JAR (there is also one for the generating define.xml), however for 3.0.1 there is only one JAR to perform all tasks.

PROCESS FLOW OF %P21_VALIDATE



TestStudy EG P21 Validation Summary Report

P21 ID	Publisher ID	Message	Severity	Count
SD1082	FDAB018	Variable length is too long for actual data	Error	13
CT2002	CG0021	EGTEST value not found in 'ECG Test Name' extensible codelist	Warning	175
CT2002	CG0021	EGTESTCD value not found in 'ECG Test Code' extensible codelist	Warning	175
CT2002	CG0021	EPOCH value not found in 'Epoch' extensible codelist	Warning	701
SD1339	FDAB022	Missing EPOCH value, when a start or observation date is provided	Warning	3

1 - CREATE A TEMPORARY FOLDER IN YOUR SAS WORK AREA

Pinnacle 21 requires SAS Version 5 (V5) transport file format (.XPT extension) datasets, which is an open standard developed by SAS and can be read by the P21 validator. In order to create the XPT versions of the CDISC dataset which we have created in our SAS session, we create a sub-folder in the temporary SAS work folder that is created on the launch of each SAS session:

```
%*--- options to not hang on windows OS;
options noxsync noxwait;


%*--- temp work path;
%let work_path = %sysfunc(getoption(work));


%*--- temp subfolder to copy XPT to;
x mkdir "&work_path\xpt";
```

```
*--- sleep SAS to allow the temp folder to be created;
data _null_;
      x=sleep(5);
run;
```

2 - CREATE A TEMPORARY XPT VERSION OF THE CDISC DATASET

When creating a transport file with the XPORT engine, you may receive a warning stating: `Engine XPORT does not support SORTEDBY operations`. As this is an expected warning with XPT files, there is no way to prevent it other than to first remove the SORT flag, then create the transport file so we included the following:

```
*--- remove sortedby to avoid warn-ing;
data &domain. (sortedby=_null_);
         set &model..&domain.;
run;
```

We then setup an XPORT library where we will create the XPT version of our dataset created by the PROC COPY:

```
libname temp_xpt xport "&work_path\xpt\&domain..xpt";

proc copy in=work out=temp_xpt;
       select &domain.;
run;
```

3 - OPEN UP P21 COMMUNITY CLI VIA SAS X COMMAND AND RUN VALIDATION

```
%*--- options to hang on windows OS;
options xsync xwait;
```

The macro was originally designed for P21C v2.2.0, but has been upgraded to be compatible with P21C v3.0.1. As the JAR modules are different this requires different programming, using the same parameters:

```
%if &p21_ver.=2.2.0 %then %do;
```

Line	SAS Code	Notes
1	`%*--- single domain validation;`	
2	`x  java -Xmx1024M -Xms1024M`	
3	`-jar \\...\tools\pinnacle21-community-2-2-0\components\lib\validator-cli-2.1.5.jar`	*Location of JAR file used by CLI*
4	`-type=&model`	***SDTM** or **ADaM***
5	`-source="&work_path\xpt\&domain..xpt"`	*Dataset to be validated specified in the call to the macro i.e. DM, ADSL*
6	`-source:type=sas`	
7	`-config="\\...\tools\pinnacle21-community-2-2-0\components\config\&model. &model_version..xml"`	***SDTM 3.2**, **ADaM 1.0***
8	`-config:cdisc=&terminology`	*NCI CT versions*
9	`-report=%sysfunc(quote(&work_path.\temp_val.xlsx))`	*Temporary P21 report to be imported to SAS*
10	`-report:type=&type.`	*Defaults to **Excel***
11	`-report:overwrite=yes`	*Defaults to **Yes***
12	`-report:cutoff=&cutoff.`	*Defaults to **1000***

```
%end;
```

An example of the resultant commands sent to the JAR via SAS to run validation:

```
java -Xmx1024M -Xms1024M -jar \\tools\pinnacle21-community-2-2-0\components\lib\validator-cli
-2.1.5.jar -type=SDTM -source="\\SAS Temporary Files\xxxxxxxx\xpt\DM.xpt" -source:type=sas -
config="P:\tools\pinnacle21-community-2-2-0\components\config\SDTM 3.2.xml" -config:cdisc=2018-
12-21 -report="J:\SAS Temporary Files\xxxxxxxx\temp_val.xlsx" -report:type=Excel -
report:overwrite=yes -report:cutoff=1000
```

```
%else %if &p21_ver.=3.0.1 %then %do;
```

Line	SAS Code	Notes
1	`%*--- single domain validation;`	
2	`x  java -Xmx1024M -Xms1024M`	
3	`%if &local.=N %then %do;`	*Option to point towards different CLI installations*
4	`-jar "\\...\Tools\pinnacle21-community-3-0-1\resources\app.asar.unpacked\components\lib\p21-client-1.0.1.jar" ^`	*Location of JAR file used by CLI – extracted to a central location*
5	`%end;`	
6	`%else %do;`	
7	`-jar "C:\Users\&sysuserid.\AppData\Local\Programs\pinnacle21-community\resources\app.asar.unpacked\components\lib\p21-client-1.0.1.jar" ^`	*Location of JAR file when installed in the default location on the users local C:\ drive*
8	`%end;`	
9	`--engine.version="&engine." ^`	***1903.1** for FDA or **1511.5** for PMDA*
10	`--standard=&model. ^`	***SDTM** or **ADaM***
11	`--standard.version=&model_version. ^`	*Version of model used i.e. **3.2** for SDTM or **1.1** for ADaM*
12	`--source.&model2.="&work_path\xpt\&domain..xpt" ^`	*Dataset to be validated specified in the call to the macro i.e. DM, ADSL – note model2 is used to ensure lower case version of &model as JAR is case specific*
13	`-cdisc.ct.&model2..version=&terminology. ^`	*NCI CT version – i.e. **2019-06-28** Note that the parameter contains a macro variable that will resolve as SDTM or JAR as:* `-cdisc.ct.sdtm.version=2019-06-28`
14	`--report="C:\Users\&sysuserid.\Desktop\temp_val.xlsx";`	*Temporary P21 report to be imported to SAS*

```
%end;
```

An example of the resultant commands sent to the JAR via SAS to run validation:

```
java -Xmx1024M -Xms1024M -jar"\\tools\pinnacle21-community-3-0-
1\resources\app.asar.unpacked\components\lib\p21-client-1.0.1.jar" ^ --engine.version="FDA
1903.1" ^ --standard=SDTM ^ --standard.version=3.2 ^ --source.sdtm="\\SAS Temporary
Files\xxxxxxxx\xpt\DM.xpt" ^ -cdisc.ct.sdtm.version=2019-06-28 ^ --
report="C:\Users\xxxxxxxx\Desktop\temp_val.xlsx"
```

4 - XSYNC/XWAIT SYSTEM OPTION FORCES SAS TO WAIT UNTIL P21 VALIDATES

After running the macro, SAS will break out to operating system using the X command and run P21 Community to validate the dataset without any further user input. It's important that XYSNC and XWAIT SAS system options are turned on so that SAS waits until the OS process has finished before returning to SAS and importing the result of the validation, as it can take up to several minutes depending on the size of the dataset being validated.

The CLI version of P21C 3.0.1 CLI requires an active internet connection for loading terminologies, dictionaries and rule implementations so is slower than P21C 2.2.0 CLI.

5 – READ VALIDATION XLSX REPORT INTO SAS

The temporary P21 validation report called `temp_val.xlsx` is then saved to the user's local desktop workspace. We found that outputting to the SAS work library caused issues due to the potential file path length, but this could be easily adjusted to save the report into the SAS temporary work folder (already assigned `&work_path`). Once read into SAS, the temporary report file is then deleted from the user's desktop using the X command.

```
%*--- import validation checks - summary;
proc import datafile="C:\Users\&sysuserid.\Desktop\temp_val.xlsx"
       out=p21_summary
       dbms=xlsx
       replace;
       sheet="Issue Summary";
       getnames=yes;
run;


%*--- import validation checks - summary;
proc import datafile="C:\Users\&sysuserid.\Desktop\temp_val.xlsx"
       out=p21_details
       dbms=xlsx
       replace;
       sheet="Details";
       getnames=yes;
run;


%*--- delete temporary validation file as has been imported back into SAS;
x %unquote(%str(%'del "C:\Users\&sysuserid.\Desktop\temp_val.xlsx"%'));
```

Some top level checks will generate by default when we validate an individual dataset i.e. *"Missing DM dataset"*, but our intention here is to only validate a single dataset then this is expected and can be dropped from the report we read back into SAS, and display to the output window. Note, as mentioned in the introduction, all datasets should be validated together during a study and we are only looking at individual datasets for an extra check.

```
%*---- keep validation checks specific to domain - drop general issues that arise due to
incomplete validation of entire study;
data p21_summary;
       set p21_summary;

       retain pinnacle_21_validator_report2;
       if pinnacle_21_validator_report ne "" then lidator_report2=pinnacle_21_validator_report;

       if pinnacle_21_validator_report2="&domain." and d ne "" then output;

       drop pinnacle_21_validator_report2;
       rename b=p21_id
              c=pub_id
              d=message
              e=severity
              f=count;

              keep b c d e f;
run;
```

6 SAS READS XLS REPORT IN AND DISPLAYS TO OUTPUT WINDOW

Depending on the macro parameters we can display the result of the validation a variety of ways to the user. The default option is to display the result of the Dataset Summary tab to the SAS window. This is most beneficial for the user at development time as it gives an overview of issues in the dataset:

print_summary=Y *(default=Y)*

TestStudy EG P21 Validation Summary Report

P21 ID	Publisher ID	Message	Severity	Count
SD1082	FDAB018	Variable length is too long for actual data	Error	13
CT2002	CG0021	EGTEST value not found in 'ECG Test Name' extensible codelist	Warning	175
CT2002	CG0021	EGTESTCD value not found in 'ECG Test Code' extensible codelist	Warning	175
CT2002	CG0021	EPOCH value not found in 'Epoch' extensible codelist	Warning	701
SD1339	FDAB022	Missing EPOCH value, when a start or observation date is provided	Warning	3

Turned off by default, we also have the option to print the full Details tab of all the individual checks that have fired:

print_details=Y (*default=N)*

TestStudy EG P21 Validation Details

Record	Variables	Values	Pinnacle 21 ID	Publisher ID	Message	Category	Severity
	EGTEST	ECG Tests	CT2002	CG0021	EGTEST value not found in 'ECG Test Name' extensible codelist	Terminology	Warning
	EGTESTCD	EGALL	CT2002	CG0021	EGTESTCD value not found in 'ECG Test Code' extensible codelist	Terminology	Warning
	EPOCH	FIRST FOLLOW-UP	CT2002	CG0021	EPOCH value not found in 'Epoch' extensible codelist	Terminology	Warning
	EPOCH	FIRST SCREENING	CT2002	CG0021	EPOCH value not found in 'Epoch' extensible codelist	Terminology	Warning
	EPOCH	FIRST TREATMENT	CT2002	CG0021	EPOCH value not found in 'Epoch' extensible codelist	Terminology	Warning
	EPOCH	SECOND SCREENING	CT2002	CG0021	EPOCH value not found in 'Epoch' extensible codelist	Terminology	Warning
	EPOCH	SECOND TREATMENT	CT2002	CG0021	EPOCH value not found in 'Epoch' extensible codelist	Terminology	Warning
	Excess, Variable	10, STUDYID	SD1082	FDAB018	Variable length is too long for actual data	Metadata	Error
	Excess, Variable	10, USUBJID	SD1082	FDAB018	Variable length is too long for actual data	Metadata	Error
	Excess, Variable	184, EGMETHOD	SD1082	FDAB018	Variable length is too long for actual data	Metadata	Error
	Excess, Variable	190, EGORRES	SD1082	FDAB018	Variable length is too long for actual data	Metadata	Error
	Excess, Variable	190, EGSTRESC	SD1082	FDAB018	Variable length is too long for actual data	Metadata	Error
	Excess, Variable	194, EGPOS	SD1082	FDAB018	Variable length is too long for actual data	Metadata	Error
	Excess, Variable	20, VISIT	SD1082	FDAB018	Variable length is too long for actual data	Metadata	Error
	Excess, Variable	24, EPOCH	SD1082	FDAB018	Variable length is too long for actual data	Metadata	Error
	Excess, Variable	26, EGTEST	SD1082	FDAB018	Variable length is too long for actual data	Metadata	Error
	Excess, Variable	3, EGTESTCD	SD1082	FDAB018	Variable length is too long for actual data	Metadata	Error
	Excess, Variable	33, EGCAT	SD1082	FDAB018	Variable length is too long for actual data	Metadata	Error
	Excess, Variable	6, DOMAIN	SD1082	FDAB018	Variable length is too long for actual data	Metadata	Error
	Excess, Variable	9, EGDTC	SD1082	FDAB018	Variable length is too long for actual data	Metadata	Error

Another option is to display the result of the checks to the log as warnings. This would be mostly useful for fully automated runs of datasets on receipt of a new data cut where log checkers would pick-up the data issues:

log_rep=Y (*default=N)*

```
WARNING: P21 Check SD0009 FDAC206 MESSAGE=No qualifiers set to 'Y', when AE is Serious SEVERITY=Error COUNT=3
WARNING: P21 Check SD0090 FDAC209 MESSAGE=AESDTH is not 'Y', when AEOUT='FATAL' SEVERITY=Error COUNT=3
WARNING: P21 Check SD0091 FDAC210 MESSAGE=AEOUT is not 'FATAL', when AESDTH='Y' SEVERITY=Error COUNT=1
WARNING: P21 Check SD1082 FDAC036 MESSAGE=Variable length is too long for actual data SEVERITY=Error COUNT=17
WARNING: P21 Check SD0021 FDAC117 MESSAGE=Missing End Time-Point value SEVERITY=Warning COUNT=1173
WARNING: P21 Check SD0031 FDAC122
MESSAGE=Missing values for AESTDTC, AESTRF and AESTRTPT, when AEENDTC, AEENRF or AEENRTPT is provided SEVERITY=Warning COUNT=3
WARNING: P21 Check SD1076 FDAC031 MESSAGE=Model permissible variable added into standard domain SEVERITY=Warning COUNT=1
NOTE: There were 7 observations read from the data set WORK.P21_SUMMARY.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

```
%if &log_rep.=Y %then %do;
    data _null_;
         set p21_summary;
             put "WARNING: P21 Check " P21_ID Pub_ID Message= Severity=
    run;
%end;
```

```
%if &print_summary.=Y %then %do;
    title1 "&study. &domain P21 Validation Summary Report";
       proc report data=p21_summary nowd headline headskip ls=150;
                column P21_ID Pub_ID Message Severity Count;
                define P21_ID          / width=15 flow "P21 ID";
                define Pub_ID          / width=15 flow "Publisher
                define message         / width=70 flow "Message";
                define severity        / width=10 flow "Severity";
                define count           / width=10 flow "Count";
       run;
%end;

%if &print_details.=Y %then %do;
    title1 "&study. &domain P21 Validation Details";
    proc report data=p21_details nowd headline headskip ls=150;
          column record variables values pinnacle_21_id publisher_id message
          define record          / width=8  flow;
          define variables       / width=20 flow;
          define values          / width=20 flow;
          define pinnacle_21_id  / width=10 flow;
          define publisher_id    / width=10 flow;
          define message         / width=30 flow;
          define category        / width=12 flow;
          define severity        / width=10 flow;
          where domain="&domain.";
    run;
%end;
```

RUNNING THE MACRO

The macro is designed to be user-friendly and can be run with a few simple parameters for example:

```
*--- simple validation of domain;
%p21_validate(study=TestStudy, model=SDTM, domain=DM);

*--- simple validation of domain and full report validation issues (not just summary) displayed;
%p21_validate(study= TestStudy, model=SDTM, domain=DM, print_details=Y);

*--- simple validation of ADaM ADSL, with warnings of issues printed to the log file;
%p21_validate(study= TestStudy, model=ADaM, domain=ADSL, log_rep=Y);
```

The full list of parameters for individual dataset validation are:

Parameter	Description
p21_ver	Version of P21 Community to use
study	Short name identifier for report file etc.
model	Specify either SDTM/ADaM to be validated
model_version	Version of the CDISC model to be validated
terminology	Version of NCI controlled terminology
log_rep	Display results of validation to log for single domain validation (default: N)
print_summary	Print the summary of validation to SAS window for single domain validation (default: Y)
print_details	Print full details of validation to SAS window for single domain validation (default: N)
type=Excel	Validation report output file type (defaulted: Excel) **(2.2.0 only)**
cutoff=1000	Repeats of data validation issues in report (defaulted: 1000)
engine	FDA is 1903.1, PMDA is 1511.5 **(3.0.1 only)**
local	Select JAR from central location or run from local installation **(3.0.1 only)**

In the standard PHASTAR study setup, there is a macro that sets the latest SDTM/ADaM models and NCI controlled terminology versions as global macro variables which **%p21_validate** uses as default values for the input parameters for **model_version** and **terminology**. This enables us to control versions on a global level. However, these values

can be overwritten to study specific versions by specifying them in the macro call directly at study level.

VALIDATION REPORT

An understanding of what each of the P21 validation checks mean is critical for efficient use of the macro i.e. when to investigate in detail the check fired or when to take into consideration the completeness of the study given checks often fire on incomplete data as the study may be ongoing and could resolve in time. The more exposure a developer gets to validation checks is beneficial long term as often it can be left to the lead study programmer to review the validation report and having developers understand how to fix, spot and avoid issues at programming level is more efficient.

Special note should be to mention that the following check will regularly fire:

FDAC036 - *"Variable length is too long for actual data"*

This is expected as only final datasets for submission should have their character variables trimmed down to the minimum required to store the complete variable and not necessarily our work version.

CONCLUSION

By adopting this macro the programming team at PHASTAR have been able to identify data, programming and specification issues within SDTM and ADaM datasets earlier in the developmental process of datasets to correct the dataset and/or specifications and catch any issues earlier to create a more efficient process.

Not discussed in this paper, the programming team regularly validate all study datasets together also using the **%p21_validate** macro which is also setup to validate all datasets together using the CLI method for complete cross-check validation. Phastar also have a related macro called **%p21_define** that generates the define.xml from the P21 specification XLSX files as input. This enables us to automate the entire process of creating the define.xml, XPTs and validating all these together programmatically within SAS whilst harnessing the power of P21 Community validator.

ACKNOWLEDGMENTS

Thanks to the programming team at PHASTAR for feedback on how to best work this macro in real world programming.

RECOMMENDED READING

X-Command: http://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/win-stmt-x.htm
Pinnacle 21 release notes on using CLI: https://www.pinnacle21.com/projects/validator/community-cli
PharmaSUG 2018 - Paper DS-20, How to Automate Validation with Pinnacle 21 Command Line Interface and SAS®: https://www.pharmasug.org/proceedings/2018/DS/PharmaSUG-2018-DS20.pdf

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

David Meek
Phastar
Unit 2A 2 Bollo Lane
London W4 5LE
Email: david.meek@phastar.com